UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/813,963	03/31/2004	Surajit Chaudhuri	307523.01	5889

22971 7590 04/05/2007
MICROSOFT CORPORATION
ONE MICROSOFT WAY
REDMOND, WA 98052-6399

EXAMINER	
HICKS, MICHAEL J	
ART UNIT	PAPER NUMBER
2165	

SHORTENED STATUTORY PERIOD OF RESPONSE	NOTIFICATION DATE	DELIVERY MODE
3 MONTHS	04/05/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Notice of this Office communication was sent electronically on the above-indicated "Notification Date" and has a shortened statutory period for reply of 3 MONTHS from 04/05/2007.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

roks@microsoft.com
ntovar@microsoft.com
a-rydore@microsoft.com

Office Action Summary	Application No.	Applicant(s)
	10/813,963	CHAUDHURI ET AL.
	Examiner	Art Unit
	Michael J. Hicks	2165

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _15 February 2007_.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1,3-45,47 and 48_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1, 3-16, 18, 21-22, 35-43, 45, and 48_ is/are rejected.

7)☒ Claim(s) _17,19,20,23-34,44 and 47_ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _15 February 2007_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1, 3-45, and 47-48 Pending.

Claims 2 and 46 Canceled.

Response to Arguments

1. Applicant's arguments filed 2/15/2007 have been fully considered but they are not persuasive.

As per Applicants argument that Eigel-Danielson fails to disclose the limitation of 'before execution of a query, defining a model of work to be performed during the execution of the query' Examiner respectfully disagrees. The cited text excerpt is indicative of the fact that a model of work is defined prior to the search. Note that model of work may reasonably be construed to mean the model by which progress will be measured. In this case, as illustrated by the text excerpt, a pre-defined model of work exists in the form of measuring by kilobytes of data to be search, or number of kilobytes needed to fill the display repository. Again note that the quoted section is pointing out that the way by which progress will be measured (e.g. the model of work) must be defined before the search begins, otherwise progress cannot be measured.

As per Applicants argument that that Eigel-Danielson fails to disclose the limitation of ' according to the model of work defined before execution of the query' Examiner respectfully disagrees. Firstly the amended claim language does not specify

that the estimating a total amount of work needs to be performed before the execution of the query, only that an estimate of the total amount of work to be performed is made. As there is no restriction as to when the estimation of total work to be done is made Examiner feels that disclosing that a total amount of work that needs to be performed is known (e.g. in order to estimate how many kilobytes are left to search, you must have an estimate as to the total amount to be searched) suffices to fulfill the claim limitation. Secondly, Examiner feels that it is clear that the work estimates are following the predefined model of work in light of the above arguments.

As per Applicants arguments that Eigel-Danielson fails to disclose that the estimating the amount of work performed at a given point during query execution and estimating the progress of the query are done according to the model of work and based on the model of work, respectively, Examiner respectfully disagrees. In light of Examiners above arguments regarding the limitation of 'before execution of a query, defining a model of work to be performed during the execution of the query', Examiner feels that the cited text excerpts clearly disclose the limitations in question. Note that the progress is expressed using percentages based off of the estimated amount of work performed and the estimated total amount of work, which are both based on the model of work.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1-3, 5, 16, 35-37, 43, 45-46, and 48 rejected under 35 U.S.C. 102(b) as being anticipated by Eigel-Danielson (U.S. Patent Number 6,301,580).

As per Claims 1 and 35, Eigel-Danielson discloses a method and computer readable medium for estimating query progress (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data..."* The preceding text excerpt clearly indicates that the progress of a query is estimated.) (Column 4, Lines 3-7), comprising: a) before execution of a query, defining a model of work to be performed during execution of the query (i.e. *"Variables can be expressed in terms of many types, such as percentages, time elapsed, and bytes of memory, and are all contemplated and within the scope of this invention. It should also be noted that the greater progress of variables can be a maximum value as well as a minimum value. For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease. Therefore, the greater progress between the two variables is the minimum value between the two variables. Likewise, if the progress of a search of a data repository is measured by the number of kilobytes of data repository already searched, and the progress of filling a display repository is measured by the number of kilobytes of display repository already filled, then*

progress is made when the two variables increase, and the greater progress between the two variables is the maximum value between the two variables." The preceding text excerpt clearly indicates that a model of work is defined (e.g. measuring the amount of work to be done in kilobytes of data to search or number of kilobytes needed to fill a display repository).) (Column 4, Lines 25-44); b) estimating a total amount of work that will be performed according to the model (i.e. *"For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease..."* The preceding text excerpt clearly indicates that the total amount of work performed is estimated (e.g. the number of kilobytes in the repository being searched is estimated).) (Column 4, Lines 29-35); c) estimating an amount of work performed according to the model at a given point during the execution of the query (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data; maintaining a second variable 502 representing the progress of filling a display repository 206 during the search inquiry, wherein the display repository 206 comprises data read from the data repository 200 and the capacity of the display repository 206 is equal to a configurable measurement of data for output..."* The preceding text excerpt clearly indicates that the amount of work performed during the execution of the query is tracked/estimated using variables.) (Column 4, Lines 3-12); d) estimating the progress of the query using the estimated amount of work performed and the estimated total amount of work (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates that the progress of the

query is expressed using percentages (e.g. the amount of work performed divided by the total work to be done).) (Column 6, Lines 25-28) and displaying estimated progress of the query to a user (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates that the progress of the query is output to the user.) (Column 6, Lines 25-28).

As per Claim 5, Eigel-Danielson discloses the work performed during execution of the query is modeled as work performed by a driver node operator during execution of the query (i.e. Note that the because applicant describes a driver node as a leaf node in a pipeline of a query execution plan (page 11 of Applicants specification, 3rd Paragraph, beginning 'Referring to Figure 5'), as the plan progresses all nodes in the query execution plan will eventually become driver nodes, thus all work performed in the query execution is done by driver nodes.).

As per Claims 16 and 43, Eigel-Danielson discloses the query progress indicator is prevented from providing an indication of decreasing query progress (i.e. *"For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease…"* The preceding text excerpt clearly indicates that because the work may be modeled as a percentage of the repository that has been searched, the progress indicator will never display a decreasing value, as the percentage of the total repository that has been search cannot decrease.) (Column 4, Lines 29-35).

As per Claim 36, Eigel-Danielson discloses a computer system including a display, a user input facility, and an application for presenting a user interface on the display (i.e. The preceding limitations are necessary for the input and processing of a query and further output of the results, and thus are considered inherent in view of the following limitations.), a user interface comprising: a) a query progress indicator that provides an indication to a user of a progress of a query (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates a query progress indicator that outputs the progress of the query to the user.) (Column 6, Lines 25-28), wherein the progress of the query is estimated (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates that the progress of the query is expressed using percentages (e.g. the amount of work performed divided by the total work to be done).) (Column 6, Lines 25-28) based on an estimated amount of work performed according to a model of work at a given point during execution of the query (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data; maintaining a second variable 502 representing the progress of filling a display repository 206 during the search inquiry, wherein the display repository 206 comprises data read from the data repository 200 and the capacity of the display repository 206 is equal to a configurable measurement of data for output..."* The preceding

text excerpt clearly indicates that the amount of work performed during the execution of the query is tracked/estimated using variables.) (Column 4, Lines 3-12) and an estimated total amount of work to be performed according to the model of work (i.e. *"For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease…"* The preceding text excerpt clearly indicates that the total amount of work performed is estimated (e.g. the number of kilobytes in the repository being searched is estimated).) (Column 4, Lines 29-35), wherein the model of work is defined before execution of the query and defines work to be performed during execution of the query (i.e. *"Variables can be expressed in terms of many types, such as percentages, time elapsed, and bytes of memory, and are all contemplated and within the scope of this invention. It should also be noted that the greater progress of variables can be a maximum value as well as a minimum value. For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease. Therefore, the greater progress between the two variables is the minimum value between the two variables. Likewise, if the progress of a search of a data repository is measured by the number of kilobytes of data repository already searched, and the progress of filling a display repository is measured by the number of kilobytes of display repository already filled, then progress is made when the two variables increase, and the greater progress between the two variables is the maximum value between the two variables."* The preceding text excerpt clearly indicates that a model of work is defined (e.g. measuring the amount of work to be done in kilobytes of data to search or number of kilobytes needed to fill a display repository).) (Column 4, Lines 25-44); and b) a query end selector that allows the user to abort execution of the query (i.e. *"In this type of scenario, valuable time and effort could be wasted because a user might ultimately cancel the search inquiry out of*

impatience and frustration." The preceding text excerpt clearly indicates that a user may cancel/abort the query the execution of a query via a query end selector.) (Column 1, Lines 39-41).

As per Claims 37 and 48, Eigel-Danielson discloses the query progress indicator provides a visual indication of a percentage of query execution that has been completed (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates a query progress indicator that outputs the progress of the query to the user in the form of a percentage of query execution that has been completed.) (Column 6, Lines 25-28).

As per Claim 45, Eigel-Danielson discloses a system for providing an indication of query progress, comprising: a) a user input device enabling a user to begin execution of a query and abort execution of a query (i.e. *"In this type of scenario, valuable time and effort could be wasted because a user might ultimately cancel the search inquiry out of impatience and frustration."* The preceding text excerpt clearly indicates that a user may cancel/abort the query the execution of a query. Also note that the fact a user may cancel a query is indicative that the user is able to input a and begin execution of a query as well.) (Column 1, Lines 39-41); b) a display; c) a data content that queries can be executed upon; d) a memory in which machine instructions are stored; e) a processor that is coupled to the user input device, to the display, to the data content, and to the memory, the processor executing the machine instructions to carry out a plurality of functions (i.e. The preceding limitations b), c), d), and e) are necessary for

the input and processing of a query and further output of the results, and thus are considered inherent in view of the following limitations.), including: i) executing a query upon the data content (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data; maintaining a second variable 502 representing the progress of filling a display repository 206 during the search inquiry, wherein the display repository 206 comprises data read from the data repository 200 and the capacity of the display repository 206 is equal to a configurable measurement of data for output..."* The preceding text excerpt clearly indicates that a query/inquiry is executed on a data repository/content.) (Column 4, Lines 3-12); ii) monitoring progress of the query (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data; maintaining a second variable 502 representing the progress of filling a display repository 206 during the search inquiry, wherein the display repository 206 comprises data read from the data repository 200 and the capacity of the display repository 206 is equal to a configurable measurement of data for output..."* The preceding text excerpt clearly indicates that the progress of the query is monitored during execution.) (Column 4, Lines 3-12); and iii) providing an indicator of query progress on the display (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is 80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. "* The preceding text excerpt clearly indicates that the progress of the query is output to the user via a display.) (Column 6, Lines 25-28), wherein the progress of the query is estimated (i.e. *"The compare function will output 506 the variable which is greater as well as any associated text, graphs, and charts that belong to that variable, which together comprise the adaptive progress indicator 508. For instance, if variable PBS is greater than PDCF, the adaptive progress indicator might read "Percentage of Buffer Searched is*

80%", where 80% is variable PBS and "Percentage of Buffer Searched" is the associated text for variable PBS. " The preceding text excerpt clearly indicates that the progress of the query is expressed using percentages (e.g. the amount of work performed divided by the total work to be done).) (Column 6, Lines 25-28) based on an estimated amount of work performed according to a model of work at a given point during execution of the query (i.e. *"The adaptive progress indicator method may comprise maintaining a first variable 500 (FIG. 5) representing the progress of a search of a data repository 200 during a search inquiry, wherein the data repository 200 comprises data; maintaining a second variable 502 representing the progress of filling a display repository 206 during the search inquiry, wherein the display repository 206 comprises data read from the data repository 200 and the capacity of the display repository 206 is equal to a configurable measurement of data for output..."* The preceding text excerpt clearly indicates that the amount of work performed during the execution of the query is tracked/estimated using variables.) (Column 4, Lines 3-12) and an estimated total amount of work to be performed according to the model of work (i.e. *"For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is made when the variables decrease..."* The preceding text excerpt clearly indicates that the total amount of work performed is estimated (e.g. the number of kilobytes in the repository being searched is estimated).) (Column 4, Lines 29-35), wherein the model of work is defined before execution of the query and defines work to be performed during execution of the query (i.e. *"Variables can be expressed in terms of many types, such as percentages, time elapsed, and bytes of memory, and are all contemplated and within the scope of this invention. It should also be noted that the greater progress of variables can be a maximum value as well as a minimum value. For instance, if the progress of a search of a data repository is measured by the number of kilobytes of data left to search the data repository, and the progress of filling a display repository is measured by the number of kilobytes needed to fill the display repository, then progress is*

made when the variables decrease. Therefore, the greater progress between the two variables is the minimum value between the two variables. Likewise, if the progress of a search of a data repository is measured by the number of kilobytes of data repository already searched, and the progress of filling a display repository is measured by the number of kilobytes of display repository already filled, then progress is made when the two variables increase, and the greater progress between the two variables is the maximum value between the two variables." The preceding text excerpt clearly indicates that a model of work is defined (e.g. measuring the amount of work to be done in kilobytes of data to search or number of kilobytes needed to fill a display repository).) (Column 4, Lines 25-44); and b) a query end selector that allows the user to abort execution of the query (i.e. *"In this type of scenario, valuable time and effort could be wasted because a user might ultimately cancel the search inquiry out of impatience and frustration."* The preceding text excerpt clearly indicates that a user may cancel/abort the query the execution of a query via a query end selector.) (Column 1, Lines 39-41).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claim 3, 6, and 38-39 rejected under 35 U.S.C. 103(a) as being unpatentable over Eigel-Danielson in view of Lezius et al. ("TigerSearch Manual", University of Stuttgart, April 5, 2002 and referred to hereinafter as Lezius).

As per Claim 3, Eigel-Danielson fails to disclose work performed during execution of a query is modeled as a number items returned by a query operator.

Lezius discloses work performed during execution of a query is modeled as a number items returned by a query operator (i.e. Figure 3.2 clearly displays a search query progress indicator which models work and results as the number of items returned by a query operator (e.g. Matching Sentences).) (Page 7, Figure 3.2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Lezius to include work performed during execution of a query is modeled as a number items returned by a query operator with the motivation of keeping the user informed about the current status of query progress (Lazius, Page 6, Heading 'Query Progress Window').

As per Claim 6, Eigel-Danielson fails to disclose work performed by a driver node operator is modeled as a number of items returned by the driver node operator.

Lezius discloses work performed by a driver node operator is modeled as a number of items returned by the driver node operator (i.e. Figure 3.2 clearly displays a search query progress indicator which models work and results as the number of items returned by a query (e.g. Matching Sentences). Note that the because applicant describes a driver node as a leaf node in a pipeline of a query execution plan (page 11 of Applicants specification, 3rd Paragraph, beginning 'Referring to Figure 5'), as the plan progresses all nodes in the query execution plan will eventually become driver nodes, thus all work performed in the query execution is done by driver nodes.) (Page 7, Figure 3.2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Lezius to include work performed by a driver node operator is modeled as a number of items returned by the driver node operator with the motivation of keeping the user informed about the current status of query progress (Lazius, Page 6, Heading 'Query Progress Window').

As per Claim 38, Eigel-Danielson discloses the percentage of query execution that has been completed is estimated by dividing the current progress of the query by an estimated total amount of work of the query (i.e. "Variables can be expressed in terms of many types, such as percentages..." The preceding text excerpt clearly indicates that the progress may be presented as a percentage of work completed (e.g. current work performed divided by total amount of work to be performed.) (Column 4, Lines 25-26).

Eigel-Danielson fails to disclose the current progress and total amount of work of the query are measured in number of items/tuples returned by the query.

Lezius discloses the current progress and total amount of work of the query are measured in number of items/tuples returned by the query (i.e. Figure 3.2 clearly displays a search query progress indicator which models work and results as the number of items returned by a query (e.g. Matching Sentences).) (Page 7, Figure 3.2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Lezius to include the current progress and total amount of work of the query are measured in number of items/tuples returned by the query with the motivation of keeping the user

informed about the current status of query progress (Lazius, Page 6, Heading 'Query Progress Window').

As per Claim 39, Eigel-Danielson discloses the percentage of query execution that has been completed is estimated by dividing the current progress of the query by an estimated total amount of work of the query (i.e. "Variables can be expressed in terms of many types, such as percentages..." The preceding text excerpt clearly indicates that the progress may be presented as a percentage of work completed (e.g. current work performed divided by total amount of work to be performed.) (Column 4, Lines 25-26).

Eigel-Danielson fails to disclose the current progress and total amount of work of the query are measured in number of items/tuples returned by an operator.

Lezius discloses the current progress and total amount of work of the query are measured in number of items/tuples returned by an operator (i.e. Figure 3.2 clearly displays a search query progress indicator which models work and results as the number of items returned by a query operator (e.g. Matching Sentences).) (Page 7, Figure 3.2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Lezius to include the current progress and total amount of work of the query are measured in number of items/tuples returned by an operator with the motivation of keeping the user informed about the current status of query progress (Lazius, Page 6, Heading 'Query Progress Window').

5. Claims 4, 7-11, 13-14, and 40-41 rejected under 35 U.S.C. 103(a) as being unpatentable over Eigel Danielson in view of Ramakrishnan et al. ("Database Management Systems", 3rd Edition, McGraw-Hill Press, 2003, Pages 404-409 and referred to hereinafter as Ramakrishnan).

As per Claim 4, Eigel-Danielson fails to disclose work performed during execution of a query is modeled as a number of GetNext( ) calls by a query operator.

Ramakrishnan discloses work performed during execution of a query is modeled as a number of GetNext( ) calls by a query operator (i.e. *"To simplify the code responsible for coordinating the execution of a plan, the relational operators that form the nodes of a plan tree (which is to be evaluated using pipelining) typically supports a uniform iterator interface, hiding the internal implementation details of each operator. The iterator interface for an operator includes the functions* ***open***, ***get_next***, *and* ***close***. *The* ***open*** *function initializes the state of the iterator by allocating buffers for its inputs and output, and is also used to pass in arguments such as selection conditions that modify the behavior of the operator. The code for the* ***get_next*** *function call the* ***get_next*** *function on each input node and calls operator specific code to process the input tuples. The output tuples generated by the processing are placed in the output buffer of the operator, and the state of the interator is updated to keep track of how much input has been consumed. When all the output tuples have been produced though repeated calls to* ***get_next***, *the* ***close*** *function is called (by the code that initiated execution of this operator) to deallocate state information."* The preceding text excerpt clearly indicates all work performed in a query operation plan, outside of allocation and deallocation of memory, is simply a series of get_next/GetNext() function calls, thus any work model used to evaluate the amount of work done in a query execution plan may be considered to include the number of get_next/GetNext() calls made.) (Page 408, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include work performed during execution of a query is modeled as a number of GetNext( ) calls by a query operator with the motivation that the get_next/GetNext() function is inherent to the operation of a query execution plan (Ramakrishnan, Page 408, Paragraphs 2-4).

As per Claim 7, Eigel-Danielson fails to disclose work performed by a driver node operator is modeled as a number of GetNext( ) calls by a driver node operator.

Ramakrishnan discloses work performed by a driver node operator is modeled as a number of GetNext( ) calls by a driver node operator (i.e. *"To simplify the code responsible for coordinating the execution of a plan, the relational operators that form the nodes of a plan tree (which is to be evaluated using pipelining) typically supports a uniform iterator interface, hiding the internal implementation details of each operator. The iterator interface for an operator includes the functions **open**, **get_next**, and **close**. The **open** function initializes the state of the iterator by allocating buffers for its inputs and output, and is also used to pass in arguments such as selection conditions that modify the behavior of the operator. The code for the **get_next** function call the **get_next** function on each input node and calls operator specific code to process the input tuples. The output tuples generated by the processing are placed in the output buffer of the operator, and the state of the interator is updated to keep track of how much input has been consumed. When all the output tuples have been produced though repeated calls to **get_next**, the **close** function is called (by the code that initiated execution of this operator) to deallocate state information."* The preceding text excerpt clearly indicates all work performed in a query operation plan, outside of allocation and deallocation of memory, is simply a series of get_next/GetNext() function calls, thus any work model used to evaluate the amount of work done in a query execution plan may be considered to include the number of get_next/GetNext() calls made. Also

note that the because applicant describes a driver node as a leaf node in a pipeline of a query execution plan (page 11 of Applicants specification, 3rd Paragraph, beginning 'Referring to Figure 5'), as the plan progresses all nodes in the query execution plan will eventually become driver nodes, thus all work performed in the query execution is done by driver nodes.) (Page 408, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include work performed by a driver node operator is modeled as a number of GetNext( ) calls by a driver node operator with the motivation that the get_next/GetNext() function is inherent to the operation of a query execution plan (Ramakrishnan, Page 408, Paragraphs 2-4).

As per Claim 8, Eigel-Danielson fails to disclose dividing a query execution plan into a set of pipelines and estimating the progress of each pipeline.

Ramakrishnan discloses dividing a query execution plan into a set of pipelines and estimating the progress of each pipeline (i.e. *"The iterator interface supports pipelining naturally; the decision to pipeline or materialize input tuples is encapsulated in the operator-specific code that processes input tuples. If the algorithm implemented for the operator allows input tuples to be processed completely when they are received, input tuples are not materialized ad the evaluation is pipelined. If the algorithm examines the same input tuple several times, they are materialized."* The preceding text excerpt clearly indicates that the iterator interface, of which the GetNext function is a part of, naturally supports pipelining of query execution plans. Note that if the query execution plan is pipelined, then each pipelines progress must be examined in order to estimate the progress of the query in whole.) (Page 408, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include dividing a query execution plan into a set of pipelines and estimating the progress of each pipeline with the motivation that the iterator interface naturally supports pipelining (Ramakrishnan, Page 408, Paragraph 4) Also note that Column 7, Lines 50-67, and Column 8, Lines 1-11 of Eigel-Danielson support the tracking of progress on multiple operations, in this case those operations could be the progress of multiple pipelines.

As per Claim 9, Eigel-Danielson fails to disclose the pipelines comprise sequences of non-blocking operators.

Ramakrishnan discloses the pipelines comprise sequences of non-blocking operators (i.e. *"The iterator interface supports pipelining naturally; the decision to pipeline or materialize input tuples is encapsulated in the operator-specific code that processes input tuples. If the algorithm implemented for the operator allows input tuples to be processed completely when they are received, input tuples are not materialized ad the evaluation is pipelined. If the algorithm examines the same input tuple several times, they are materialized."* The preceding text excerpt clearly indicates that a pipeline is defined as a series of non-blocking operators (e.g. operators which will not make multiple calls to the same input tuple, therefor blocking access to that tuple for at least on of the operators).) (Page 408, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include the pipelines comprise sequences of non-blocking operators

with the motivation of that the iterator interface naturally supports pipelining (Ramakrishnan, Page 408, Paragraph 4).

As per Claim 10, Eigel-Danielson fails to disclose combining progress estimates for the pipelines to estimate the progress of the query.

Ramakrishnan discloses combining progress estimates for the pipelines to estimate the progress of the query (i.e. *"The iterator interface supports pipelining naturally; the decision to pipeline or materialize input tuples is encapsulated in the operator-specific code that processes input tuples. If the algorithm implemented for the operator allows input tuples to be processed completely when they are received, input tuples are not materialized ad the evaluation is pipelined. If the algorithm examines the same input tuple several times, they are materialized."* The preceding text excerpt clearly indicates that the iterator interface, of which the GetNext function is a part of, naturally supports pipelining of query execution plans. Note that if the query execution plan is pipelined, then each pipelines progress must be examined and combined in order to estimate the progress of the query in whole.) (Page 408, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include combining progress estimates for the pipelines to estimate the progress of the query with the motivation that the iterator interface naturally supports pipelining (Ramakrishnan, Page 408, Paragraph 4).

As per Claim 11, Eigel-Danielson fails to disclose initializing an estimate of the total amount of work that will be performed by a pipeline with an estimate from a query optimizer.

Ramakrishnan discloses initializing an estimate of the total amount of work that will be performed by a pipeline with an estimate from a query optimizer (i.e. *"Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan. The optimizer generates alternative plans and chooses the plan with the least estimated cost."* The preceding text excerpt clearly indicates that all query execution plans will be costed before execution, and an initial estimate of the cost is generated. In light of the above disclose, in order to generate an estimated cost for a pipelined query, the estimated cost of each pipeline must be generated.) (Page 404, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include initializing an estimate of the total amount of work that will be performed by a pipeline with an estimate from a query optimizer with the motivation that query costing is inherent to the operation of a query optimizer, and all queries are presented to a query optimizer before execution (Page 404, Paragraphs 5-6).

As per Claim 13, Eigel-Danielson fails to disclose identifying driver node operators of the pipeline and modeling the work performed during execution of the pipelines as work performed by the driver node operators

Ramakrishnan discloses identifying driver node operators of the pipeline and modeling the work performed during execution of the pipelines as work performed by the driver node operators (i.e. *"Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan. The optimizer generates alternative plans and chooses the plan with the least estimated cost."* The preceding text excerpt clearly indicates that all query execution plans will be costed before execution, and an initial estimate of the cost is generated. In

light of the above disclose, in order to generate an estimated cost for a pipelined query, the estimated cost of each pipeline must be generated (the estimate costs for each pipeline including the driver node for that pipeline), and the estimated for each pipeline (and corresponding driver node) must be combined to get the total cost (e.g. amount of work to be performed) of executing the query.) (Page 404, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include identifying driver node operators of the pipeline and modeling the work performed during execution of the pipelines as work performed by the driver node operators with the motivation that query costing is inherent to the operation of a query optimizer, and all queries are presented to a query optimizer before execution (Page 404, Paragraphs 5-6).

As per Claim 14, Eigel-Danielson fails to disclose modeling the work performed during execution of the pipelines as work performed by all operators in the pipeline.

Ramakrishnan discloses modeling the work performed during execution of the pipelines as work performed by all operators in the pipeline (i.e. *"Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan. The optimizer generates alternative plans and chooses the plan with the least estimated cost."* The preceding text excerpt clearly indicates that all query execution plans will be costed before execution, and an initial estimate of the cost is generated. In light of the above disclose, in order to generate an estimated cost for a pipelined query, the estimated cost of each pipeline must be generated, and the estimated for each pipeline must be combined to get the total cost (e.g. amount of work to be performed) of executing the query.) (Page 404, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include modeling the work performed during execution of the pipelines as work performed by all operators in the pipeline with the motivation that query costing is inherent to the operation of a query optimizer, and all queries are presented to a query optimizer before execution (Page 404, Paragraphs 5-6).

As per Claim 40, Eigel-Danielson discloses the percentage of query execution that has been completed is estimated by dividing the current progress of the query by an estimated total amount of work of the query (i.e. "Variables can be expressed in terms of many types, such as percentages..." The preceding text excerpt clearly indicates that the progress may be presented as a percentage of work completed (e.g. current work performed divided by total amount of work to be performed.) (Column 4, Lines 25-26).

Eigel-Danielson fails to disclose the current progress and total amount of work of the query are measured in number of GetNext() calls performed.

Ramakrishnan discloses the current progress and total amount of work of the query are measured in number of GetNext() calls performed (i.e. *"To simplify the code responsible for coordinating the execution of a plan, the relational operators that form the nodes of a plan tree (which is to be evaluated using pipelining) typically supports a uniform iterator interface, hiding the internal implementation details of each operator. The iterator interface for an operator includes the functions* ***open****,* ***get_next****, and* ***close****. The* ***open*** *function initializes the state of the iterator by allocating buffers for its inputs and output, and is also used to pass in arguments such as selection conditions that modify the behavior of the operator. The code for the* ***get_next*** *function call the* ***get_next*** *function on each input node and calls operator specific code to process the input tuples. The output tuples generated*

*by the processing are placed in the output buffer of the operator, and the state of the interator is updated to keep track of how much input has been consumed. When all the output tuples have been produced though repeated calls to **get_next**, the **close** function is called (by the code that initiated execution of this operator) to deallocate state information."* The preceding text excerpt clearly indicates all work performed in a query operation plan, outside of allocation and deallocation of memory, is simply a series of get_next/GetNext() function calls, thus any work model used to evaluate the amount of work done in a query execution plan may be considered to include the number of get_next/GetNext() calls made.) (Page 408, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include the current progress and total amount of work of the query are measured in number of GetNext() calls performed with the motivation the get_next/GetNext() function is inherent to the operation of a query execution plan (Ramakrishnan, Page 408, Paragraphs 2-4).

As per Claim 41, Eigel-Danielson fails to disclose initializing the estimated total number of GetNext( ) calls with an estimate from a query optimizer.

Ramakrishnan discloses initializing the estimated total number of GetNext( ) calls with an estimate from a query optimizer (i.e. *"Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan. The optimizer generates alternative plans and chooses the plan with the least estimated cost."* The preceding text excerpt clearly indicates that all query execution plans will be costed before execution, and an initial estimate of the cost is generated. In light of the above disclose, the GetNext call is an inherent part of a query execution plan model and will thus be used in the costing of a query execution plan.) (Page 404, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include initializing the estimated total number of GetNext( ) calls with an estimate from a query optimizer with the motivation of query costing is inherent to the operation of a query optimizer, and all queries are presented to a query optimizer before execution (Page 404, Paragraphs 5-6).

6. Claim 12 rejected under 35 U.S.C. 103(a) as being unpatentable over Eigel-Danielson in view of Kabra et al. (Reference C in Applicants IDS dated March 25, 2005 and referred to hereinafter as Kabra).

As per Claim 12, the Eigel-Danielson fails to disclose refining the initial estimate of the total work using feedback obtained during query execution.

Kabra discloses refining the initial estimate of the total work using feedback obtained during query execution (i.e. *"In this paper, we describe Dynamic Re-Optimization, an algorithm that can detect the sub-optimality of a query execution plan while executing the query in order to re-optimize it and improve its performance. During query optimization, the plan produced by the query optimizer is annotated with the various estimates and statistics used by the optimizer. Actual statistics are collected at query execution time. These observed statistics are compared against the estimated statistics and the difference is taken as an indicator of whether the query-execution plan is sub-optimal. The new statistics (much more accurate than the initial optimizer estimates) can now be used to optimize the execution of the remainder of the query."* The preceding text excerpt clearly indicates that statistics/feed back collected during query execution are used to refine the estimation of total work (e.g. a determination of whether the execution plan is sub-optimal is made).) (Page 106, Column 2, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson and Ramakrishnan with the teachings of Kabra to include refining the initial estimate of the total work using feedback obtained during query execution with the motivation of re-optimizing, and thus re-costing queries during execution (Kabra, Abstract).

7. Claims 15, 18, 21-22, and 42 rejected under 35 U.S.C. 103(a) as being unpatentable over Eigel Danielson in view of Ramakrishnan and further in view of Kabra.

As per Claim 15, Eigel-Danielson fails to disclose identifying driver node operators of the pipeline and using information about the driver node operators obtained during execution to estimate a total amount of work that will be performed by all operators in the pipeline.

Ramakrishnan discloses identifying driver node operators of the pipeline and using information about the driver node operators to estimate a total amount of work that will be performed by all operators in the pipeline (i.e. *"Queries are parsed and then presented to a query optimizer, which is responsible for identifying an efficient execution plan. The optimizer generates alternative plans and chooses the plan with the least estimated cost."* The preceding text excerpt clearly indicates that all query execution plans will be costed before execution, and an initial estimate of the cost is generated. In light of the above disclose, in order to generate an estimated cost for a pipelined query, the estimated cost of each pipeline must be generated (the estimate costs for each

pipeline including the driver node for that pipeline), and the estimated for each pipeline (and corresponding driver node) must be combined to get the total cost (e.g. amount of work to be performed) of executing the query.) (Page 404, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include identifying driver node operators of the pipeline and using information about the driver node operators obtained to estimate a total amount of work that will be performed by all operators in the pipeline with the motivation that query costing is inherent to the operation of a query optimizer, and all queries are presented to a query optimizer before execution (Page 404, Paragraphs 5-6).

Ramakrishnan fails to disclose that the information is obtained during execution.

Kabra discloses that the information is obtained during execution (i.e. *"In this paper, we describe Dynamic Re-Optimization, an algorithm that can detect the sub-optimality of a query execution plan while executing the query in order to re-optimize it and improve its performance. During query optimization, the plan produced by the query optimizer is annotated with the various estimates and statistics used by the optimizer. Actual statistics are collected at query execution time. These observed statistics are compared against the estimated statistics and the difference is taken as an indicator of whether the query-execution plan is sub-optimal. The new statistics (much more accurate than the initial optimizer estimates) can now be used to optimize the execution of the remainder of the query."* The preceding text excerpt clearly indicates that statistics/feed back collected during query execution are used to refine the estimation of total work (e.g. a determination of whether the execution plan is sub-optimal is made).) (Page 106, Column 2, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Kabra to

include that the information is gathered during execution with the motivation of re-optimizing, and thus re-costing queries during execution (Kabra, Abstract).

As per Claim 18, the Eigel-Danielson and Ramakrishnan fail to disclose identifying a spill of tuples during query execution and adjusting the model of work to account for additional work that results from the spill of tuples.

Kabra discloses identifying a spill of tuples during query execution and adjusting the model of work to account for additional work that results from the spill of tuples (i.e. *"At query execution time, the Memory Manager of the database engine determines the allocation of memory to the various operators of the query. It determines the memory requirements (minimum and maximum memory demands) of each operator using the estimates provided by the optimizer. Based on the memory requirements of each operator, and by considering the trade-offs involved, it allocates some amount of memory to each operator. The amount of memory thus allocated to an operator represents the maximum memory that the operator is allowed to use during execution. If all the data required by the operator does not fit into the allocated amount of memory, it has to spill some of the data to disk. Details of the Memory Management module of Paradise are described in [15]."* The preceding text excerpt clearly indicates that tuple spill is monitored and taken into account in the implementation of the query re-optimizer.) (Page 113, Column 2, Paragraph 5; Page 114, Column 1, Paragraph 1).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson and Ramakrishnan with the teachings of Kabra to include identifying a spill of tuples during query execution and adjusting the model of work to account for additional work that results from the spill of tuples with the motivation that tuple spill is monitored during query execution (Kabra, Page 114, Column 1, Paragraph 1).

As per Claim 21, the Eigel-Danielson and Ramakrishnan fail to disclose updating an estimated total amount of work that will be performed during query execution.

Kabra discloses updating an estimated total amount of work that will be performed during query execution (i.e. *"In this paper, we describe Dynamic Re-Optimization, an algorithm that can detect the sub-optimality of a query execution plan while executing the query in order to re-optimize it and improve its performance. During query optimization, the plan produced by the query optimizer is annotated with the various estimates and statistics used by the optimizer. Actual statistics are collected at query execution time. These observed statistics are compared against the estimated statistics and the difference is taken as an indicator of whether the query-execution plan is sub-optimal. The new statistics (much more accurate than the initial optimizer estimates) can now be used to optimize the execution of the remainder of the query."* The preceding text excerpt clearly indicates that statistics/feed back collected during query execution are used to update the estimation of total work (e.g. a determination of whether the execution plan is sub-optimal is made).) (Page 106, Column 2, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson and Ramakrishnan with the teachings of Kabra to include updating an estimated total amount of work that will be performed during query execution with the motivation of re-optimizing, and thus re-costing queries during execution (Kabra, Abstract).

As per Claim 22, the Eigel-Danielson and Ramakrishnan fail to disclose an estimated amount of work performed according to the model is updated at a plurality of points during query execution.

Kabra discloses an estimated amount of work performed according to the model is updated at a plurality of points during query execution (i.e. *"At specific intermediate points in the query, various statistics are collected during query execution. These statistics are used to obtain improved estimates of the sizes of intermediate results and execution costs. These improved estimates can be compared against the optimizer's estimates to detect sub-optimality of the query execution plan."* The preceding text excerpt clearly indicates that the updating of estimated work may be done a plurality of points during query execution.) (Page 107, Column 1, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson and Ramakrishnan with the teachings of Kabra to include an estimated amount of work performed according to the model is updated at a plurality of points during query execution with the motivation of re-optimizing, and thus re-costing queries during execution (Kabra, Abstract).

As per Claim 42, the Eigel-Danielson fails to disclose initial estimate of the total number of GetNext( ) calls is updated using feedback obtained during query execution.

Kabra discloses initial estimate of the total work to be performed is updated using feedback obtained during query execution (i.e. *"In this paper, we describe Dynamic Re-Optimization, an algorithm that can detect the sub-optimality of a query execution plan while executing the query in order to re-optimize it and improve its performance. During query optimization, the plan produced by the query optimizer is annotated with the various estimates and statistics used by the optimizer. Actual statistics are collected at query execution time. These observed statistics are compared against the estimated statistics and the difference is taken as an indicator of whether the query-execution plan is sub-optimal. The new statistics (much more accurate than the initial optimizer estimates) can now be used to optimize the execution of the remainder of the query."* The preceding text excerpt clearly

indicates that statistics/feed back collected during query execution are used to update the estimation of total work (e.g. a determination of whether the execution plan is sub-optimal is made).) (Page 106, Column 2, Paragraph 4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Kabra to include initial estimate of the total work to be performed is updated using feedback obtained during query execution with the motivation of re-optimizing, and thus re-costing queries during execution (Kabra, Abstract).

Kabra fails to disclose that the estimated total work performed by the query is measure in the number of GetNext() calls performed by the query.

Ramakrishnan discloses the total amount of work of the query is measured in number of GetNext() calls performed (i.e. *"To simplify the code responsible for coordinating the execution of a plan, the relational operators that form the nodes of a plan tree (which is to be evaluated using pipelining) typically supports a uniform iterator interface, hiding the internal implementation details of each operator. The iterator interface for an operator includes the functions* ***open***, ***get_next***, *and* ***close***. *The* ***open*** *function initializes the state of the iterator by allocating buffers for its inputs and output, and is also used to pass in arguments such as selection conditions that modify the behavior of the operator. The code for the* ***get_next*** *function call the* ***get_next*** *function on each input node and calls operator specific code to process the input tuples. The output tuples generated by the processing are placed in the output buffer of the operator, and the state of the interator is updated to keep track of how much input has been consumed. When all the output tuples have been produced though repeated calls to* ***get_next***, *the* ***close*** *function is called (by the code that initiated execution of this operator) to deallocate state information."* The preceding text excerpt clearly indicates all work performed in a query operation plan, outside of allocation and deallocation of memory, is simply a series of get_next/GetNext()

function calls, thus any work model used to evaluate the amount of work done in a query execution plan may be considered to include the number of get_next/GetNext() calls made.) (Page 408, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Eigel-Danielson with the teachings of Ramakrishnan to include the current progress and total amount of work of the query are measured in number of GetNext() calls performed with the motivation the get_next/GetNext() function is inherent to the operation of a query execution plan (Ramakrishnan, Page 408, Paragraphs 2-4).

Allowable Subject Matter

8. Claims 17, 19-20, 23-34, 44, and 47 objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

The following is a statement of reasons for the indication of allowable subject matter: The prior art of record neither teaches nor suggests the limitations of using an upper bound of a total work estimation for a query to prevent a query progress indicator to indicate decreasing progress, maintaining upper and lower bounds on the total amount of work to be performed or items that will be returned by the query operator and modifying the estimated work totals when the item values fall outside the bounds, a tuple spill indicator associated with a query progress bar indicator, or assigning weights to the pipelines.

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Points of Contact

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Friday 10:00a - 7:00p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Gaffin can be reached on (571) 272-4146. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Michael J Hicks
Art Unit 2165
Phone: (571) 272-2670
Fax: (571) 273-2670

JEFFREY GAFFIN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100